Journal of Microscopy, *Vol.* 148, *Pt* 1, *October* 1987, *pp.* 51–87.
Received 15 *October* 1986; revised 3 *May* 1987; *accepted* 18 *May* 1987

A LISP-based image analysis system with applications to microscopy

by DAVID S. BRIGHT, *Chemistry A121, National Bureau of Standards, Gaithersburg, MD 20899, U.S.A.*

KEY WORDS. Image analysis, image analysis system, image processing, image processing algorithms, LISP, microanalysis.

SUMMARY

LISPIX is an image processing and analysis system consisting of a sub-system of image processing commands implemented in FORTRAN and a collection of analysis algorithms implemented in LISP. Examples are given of using the system for microanalysis applications. The processing commands are useful alone for simple tasks. For analysis applications, the LISP programming environment, for which the processing commands are designed, allows rapid development of algorithms and a convenient way to use them. Appendices contain definitions of the most useful commands as well as examples of the LISP algorithms.

LISPIX: A LISP-BASED IMAGE ANALYSIS SYSTEM WITH APPLICATIONS TO MICROSCOPY

The need for LISPIX

In microbeam analysis, chemical morphological and structural information resides in images obtained from various instruments such as the analytical electron microscope, the electron microprobe and the ion microscope. We wish to automate the recognition of various objects or regions of interest in these images and then automate measurements on these regions.

Algorithms to perform object recognition are complex and under continual development. An image analysis system needs on the one hand to be able to handle these complex algorithms and data structures with reasonable facility, and on the other hand, to provide for large two-dimensional arrays (the images) and the fast, efficient algorithms that operate on these arrays.

For our microanalysis applications, we have developed a special hybrid system named LISPIX after its historical counterpart (Orser *et al.*, 1972). The major new feature of LISPIX is the use of LISP for system control. LISPIX is implemented on our VAX* computer and consists of a FORTRAN image processing system driven by LISP functions. LISP is a powerful information processing language that was developed in the 1950s for symbolic processing, and is used by the Artificial Intelligence community for such

*The commercial equipment identified in this paper serves only to more fully describe the subject discussed. In no case does this identification imply recommendation by the U.S. Government.

applications as expert systems and vision research (Winston & Horn, 1984). It is designed for processing symbols, as well as numbers, and for programming with lists of symbols numbers and complex lists of all of these—thus the name LISt Programming or LISP. LISP presents a good environment for algorithm development and helps us to be software compatible with those in Artificial Intelligence research.

LISPIX is designed for both image processing and image analysis. Image processing results in another image for visual inspection or analysis, while image analysis results in numeric or symbolic data describing the image. Image processing is often performed with the FORTRAN part alone (one command at a time), while image analysis is done with the entire system, making extensive use of LISP functions. Image analysis may involve measuring properties such as the size distribution of a field of particles or such as determining the basis vectors that describe the spacings of electron diffraction spots. This type of information is often useful if displayed graphically, so both parts of the LISPIX system have graphics capability.

The LISPIX system

The two parts of LISPIX serve two distinct purposes (see Fig. 1). The first (FORTRAN) part stores the images as arrays of integers and operates directly on them by interpreting and executing the LISPIX commands. The second (LISP) part controls the FORTRAN part, works with more abstract information, and is optional. Typing commands on the keyboard is useful for some simple image processing tasks. LISP is used routinely to automatically send commands for data collection and analysis.

The first part consists mainly of the commands, hereafter called 'LISPIX commands'. These LISPIX commands are incorporated into a simple image oriented programming

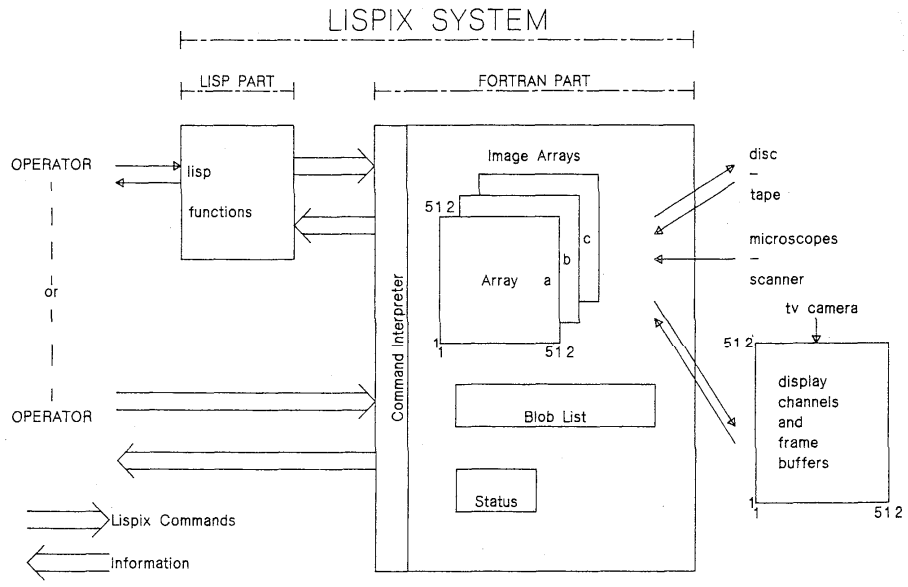

Fig. 1. Block diagram of LISPIX system. LISPIX arrays are layered on each other, as far as the comman are concerned. Coordinate systems are the same for arrays and display. The blob list is described Appendix I under 'Image Analysis Commands'. The status information includes the default bounding rectangle, the bounding rectangle of the image last read, the default image directory, and histories of the images in each array.

language or sub-system. This paper describes the commands in detail and gives examples of their use. The commands manipulate the images as arrays of integers, pixel by pixel. The commands operate on one or more of the arrays and result in one or more images in other arrays. In these respects the commands are similar to other image processing systems (Tamura *et al.*, 1982; Landy *et al.*, 1984; Meinzer & Engelmann, 1984).

The syntax of the commands and the type of information returned are important as they specify the communication between the two parts of the system. Since each part is large or complex enough to warrant its own process (or computer) it is very helpful to have only short textual messages being passed between them. The division of the image analysis task into two parts also has the advantage of allowing each part to be performed by a computer language well suited to that part.

The syntax of a command consists of the command name followed by a simple list of parameters, as described in the command overview in Appendix I. This system is adaptable enough to facilitate adding new commands and simple enough to be easy to use. The command interpreter (Fig. 1) merely uses spaces to divide the command into its parts, and ignores parentheses to make LISP control more convenient as well as to minimize typing for keyboard control. The sub-system returns information from various commands in list form which can be read either by an operator or by LISP functions. This is described in more detail in the definitions of the commands given in Appendix II.

While some image processing operations are performed with single commands, all of the image analysis algorithms are written as collections of LISP functions, each of which may use several LISPIX commands. These functions make rare the times when an operator must be concerned with explicit coordinates or specific image arrays. Some of the LISP functions use commands to process an image, extract information or do graphics. The other functions usually work in the LISP environment to process the information. A few of the LISP functions used in the examples are given in Appendix III, but most of the LISP functions are not described in this paper because they are specific to a particular application and will appear in subsequent papers.

Advantages

Some other image processing systems are file based (Landy *et al.*, 1984). They read an image with history from a file, do operations, add the operations to the history, and store the image and the new history. This is ideal for applications where the histories are routine, uniform or relatively short, and the desired result is again an image. LISPIX can mimic this process by generating a series of commands.

A series of fixed commands is adequate if no information from any command (processing step) is needed for any subsequent command. However, algorithms for analysis often use results from one processing step to guide the next. The image histories from these algorithms (as in Examples 5 and 8 below) often become long and difficult to read. Due to the power and expressiveness of LISP, the LISP functions themselves are the most compact way to represent the analysis algorithms.

Inputs and outputs

Images are the input data for LISPIX. Our images are monochrome pictures represented in the computer as two-dimensional numeric arrays. Each element or number in the array corresponds to a pixel or small spot of the image. As shown in Fig. 1, a pixel at location (i,j) corresponds to an element of one of the arrays, such as $b(i,j)$.

Single commands often involve the simultaneous use of two or more arrays, and analysis algorithms often use many commands. Thus it is advantageous to have a number of images in memory at once. Depending on the application, our LISPIX image configurations are typically six 512×512 pixel images or ninety-six 128×128 pixel images. The latter configuration is used on some projects where many images are compared in one operation,

such as when calculating the median, pixel by pixel, of a group of images, or calculating the Student's *t* score for pixel by pixel differences in two groups of images.

Although each pixel corresponds to a two-byte (sixteen bit) integer, the real dynamic range for each pixel depends on the image source (see below). Individual images are monochrome in the usual sense in that we do not have colour cameras that give three monochrome images for any field of view. However, with several of our instruments, a single field of view can produce many monochrome images, each corresponding to a different imaging mode or to a different chemical element.

The images are transferred to LISPIX via the image processor or by disc files. Our image sources are:

(a) *Analytical electron microscope.* The analytical electron microscope (Hren *et al.*, 1979) a hybrid of the transmission electron microscope (TEM), scanning transmission electron microscope (STEM) and scanning electron microscope (SEM), produces both scanned and direct (optical) images. Direct images in the TEM image modes give high resolution morphology and crystal structure. Scanned images provide morphology and crystal structure with signals from electron detectors, while X-ray and electron energy loss spectrometers provide signals for elemental maps. Digital images for all but the TEM mode are obtained directly from the computer that controls the microscope and the multiplexed multichannel analyser that is connected to the above detectors for simultaneous imaging in several different modes. These images are usually 512×512 one byte pixels. TEM mode images are obtained from a television camera focused on the phosphor. These images are usually 512×512 one byte pixels.

(b) *Television cameras.* Images from various television cameras are digitized by the DeAnza IP8400 image processor. The images are 512×512 one or two byte pixels. The second byte of precision comes from a summation of 256 TV frames, each with one byte pixels. Typically we retain one byte of precision for light microscope bright field images and two bytes of precision for ion microscope images.

(c) *Electron probe.* The electron probe microanalyser produces scanning images with various signals (secondary electrons, backscattered electrons and absorbed current) to reveal specimen physical structure, and characteristic X-ray signals to reveal specimen elemental composition (Goldstein *et al.*, 1981; Heinrich, 1981). Pixel values are X-ray count rates. There are always several images per sample, each image representing the count rates for a given element. Both energy dispersive and wavelength dispersive detectors are used. Because of the time required to get an adequate number of counts for each pixel, the images are usually 256×256 pixels or smaller. The pixels have two bytes of precision.

(d) *Optical drum scanner.* This instrument scans negatives with a small spot of light, yielding high resolution images of various sizes, with one byte pixels.

(e) *Ion microscope.* The direct imaging ion microscope, a form of secondary ion mass spectrometer, produces mass-separated ion images on a fluorescent screen which follows a channel plate. A television camera views this screen.

Examples of LISPIX applied to microscope images

All of these examples are executed with LISP functions. Some are simple enough to be done with the LISPIX commands directly, but even then the LISP functions serve as documentation of what was done, and save substantial typing by the operator. Some of these LISP functions are described in Appendix III.

(1) *Blemish elimination in electron probe blanks.* Electron probe microanalyser X-ray images show a pronounced defocusing effect at low magnification (Marinenko *et al.*, 1986). The instrument thus has a sensitivity that is non-uniform over the field of view, as illustrated in Fig. 2, which is a 128×128 pixel X-ray map of a pure chromium surface. The contrast has been enhanced for display purposes. The very dark spots and lines are caused by pits and scratches. Except for these defects, this image could be used to correct the

sensitivity for chromium concentration measurements in data from samples studied at the same instrument settings. This correction is done by dividing the sample images by the pure chromium image to form a '*k* value' array for subsequent matrix correction. It is difficult to obtain a blemish-free surface on all occasions, but if the blemishes are sufficiently small or thin, they can be nearly eliminated with the median filter.

The change in sensitivity should be a smoothly varying function of position in the image, seen as the bright diffuse stripe that travels across the image from the lower left corner to the upper right. The darker rectangular region to the upper left is caused by contamination from a previous raster scan. Normally, another region would be chosen as a

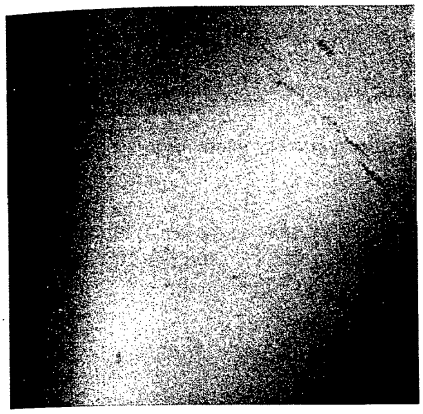

Fig. 2

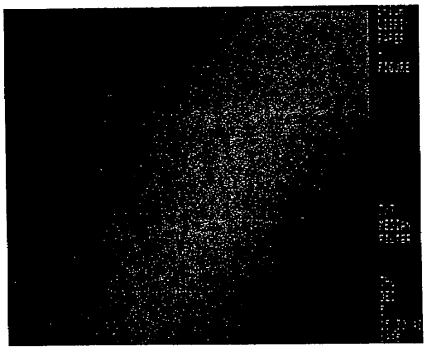

Fig. 3

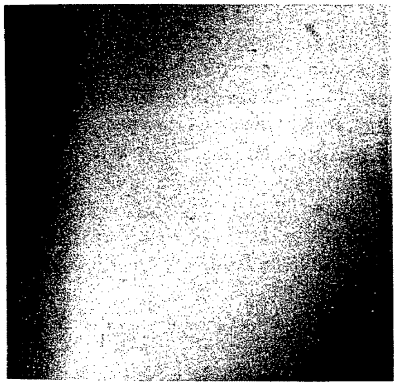

Fig. 4

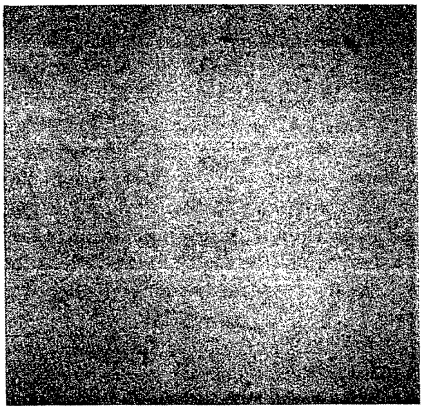

Fig. 5

Fig. 2. Chromium X-ray map, 128×128 pixels, polished chromium standard. Large square darker region at the upper left: contamination from previous raster scan. Lines and spots: scratches and pits on surface.

Fig. 3. Figure 22 filtered with 7×7 median filter. Surface defects removed, edges of previous raster scan preserved. Annotation corresponds to Appendix III, Example 1.

Fig. 4. Figure 2 filtered with 7×7 mean filter (normal smoothing). Defects are still visible, and edges of raster scan are blurred.

Fig. 5. Transmission light micrograph of asbestos fibres, with no contrast enhancement.

standard region to avoid this area, but it is included here for comparing the characteristics of the mean and median filters.

Figures 3 and 4 show the results of smoothing the original with a 7×7 median filter and a 7×7 mean filter. The median filter has the desirable results of eliminating the surface defects while leaving the larger features intact including the sharp edges of the raster scan contaminated region to the upper left. The mean filter removes the pits but leaves a residual scratch that is twice as dark as that left by the median filter.

The contrast of all of the images has been enhanced to the same extent. The real intensities just inside and outside of the dark raster scan to the upper left differ by about 5%. As might be expected for relatively constant regions, the mean and median filtered images for these regions as well as other regions outside the defects are essentially unchanged: the averages of representative areas differ from the original by less than 0·4%.

(2) *Background correction—transmission optical microscopy.* If a blank image is available for transmission light or electron microscopic images, the sample images can be corrected for uneven background illumination and the camera sensitivity factor by dividing by the blank image.

Assuming linear detector response and using Beer's law for each pixel,

$$I=L*A*S \tag{a}$$

where I is the pixel intensity, L is the illumination for that pixel light path, A is the fraction of light transmitted by the object and S is the sensitivity factor of the (linear) detector.

With no sample present, the pixel intensity, I_b, for the blank image is:

$$I_b=L*S \tag{b}$$

Dividing Eq. (a) by Eq. (b) gives:

$$A=I/I_b \tag{c}$$

A is the desired intensity, to within a scaling factor that is constant for all pixels. The sample image may thus be corrected for variations in background illumination and detector sensitivity by dividing by the blank image.

Figure 5 is a transmission light micrograph of asbestos fibres, which is nearly white as the image has low contrast and has not been contrast enhanced. Often it is desirable to set the threshold for such images to select the objects, as has been done in Fig. 6. This figure emphasizes the problem that uneven background illumination poses for analysis: the background at the edges is darker than the objects at the centre so that objects near the edges are completely missing. Figure 7 is the micrograph in Fig. 5 with the contrast enhanced to show the uneven background illumination. Figure 8 is the corrected image obtained by dividing Fig. 5 by a blank image, with the contrast enhanced as with Fig. 7. Figure 9 is the corrected image with the threshold set to select the fibres in the same way as

---

Fig. 6. Figure 5 with the threshold set to select the fibres. Fibres at periphery are lost due to uneven background illumination.

Fig. 7. Figure 5, contrast enhanced to show uneven background illumination.

Fig. 8. Figure 5, corrected by dividing by a blank image taken from a clear area of the microscope slide. Contrast enhanced.

Fig. 9. Figure 8 with threshold set to select fibres. Fibres are selected correctly (contrast with Fig. 6).

Fig. 10. Figure 5, corrected as with Fig. 8, except blank range image smoothed first to eliminate small blemishes due to dust on clear area of microscope slide. Compare small spots in the background of this image with those in Fig. 8 (see text).

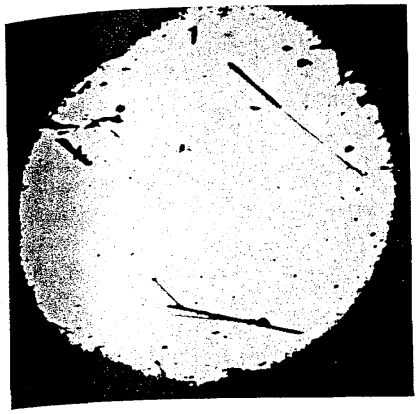

Fig. 6

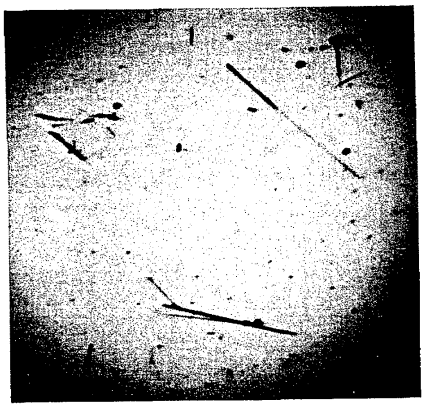

Fig. 7

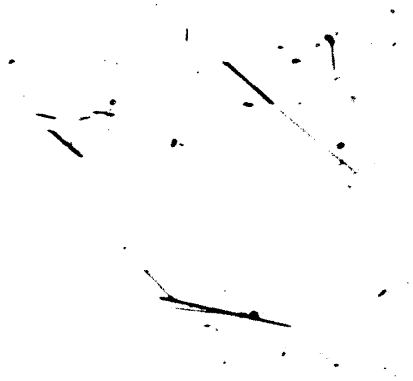

Fig. 8

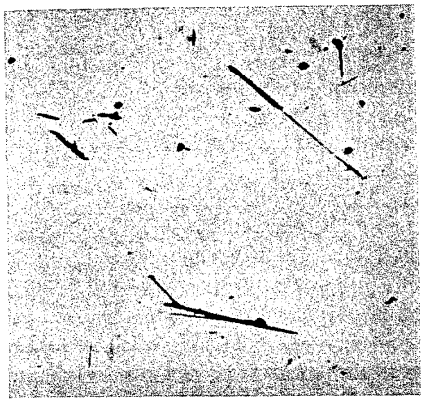

Fig. 9

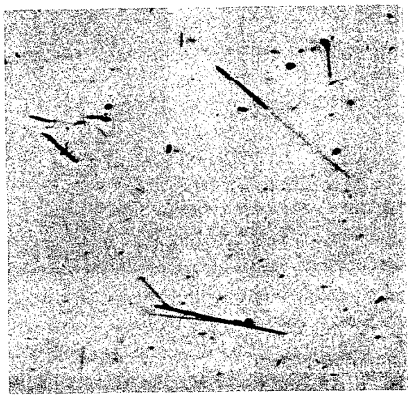

Fig. 10

Fig. 6. Fibres at the edges of the field of view are now selected along with those at the centre.

It was difficult to obtain a perfect blank image. The one used in getting Fig. 8 had dark spots due to camera defects. A slight misregistration, probably caused by small changes in the focusing optics, offset the dark spots in the blank image and caused the artificially bright spots in the corrected image (Fig. 8). If the blank image is smoothed, in this case with a 17×17 median filter, then the bright spots are eliminated in the corrected image, as shown in Fig. 10. The trade off is that the spots that also appear in the blank image are not removed in this figure, while they are removed from Fig. 8.

Another example of correcting background illumination problems before setting the threshold to select objects is illustrated by Figs 11 and 12. These are processed images of a transmission light micrograph of a reticle intended for size calibration. We want to select the lines for application of the Hough transform, which maps lines into points (see Example 7). Figure 11 is the original micrograph, not corrected for background illumination, with the threshold set in an attempt to isolate all of the lines. Figure 12 shows the properly isolated lines of the image corrected for background illumination, again with the threshold set as in Fig. 11. An experimental background image was not available in this case, so one was generated from the original image by using a maximum (rank=100%, see the RANK_FILTER command in Appendix III) filter with a kernel size that was a few reticle line spacings wide.

(3) *Spot selection.* The top hat filter (Bright & Steel, 1987; also see Appendices I and II) is useful for discriminating spots from extended objects, such as the rings and diffuse background in Fig. 13. This is an electron diffraction pattern, as photographed from the phosphor screen and digitized from the negative with the optical drum scanner. In order to measure the spot locations for crystallographic measurements, the spots need to be isolated from the rest of the image. Figure 14 shows the result of applying the top hat filter to this image, with parameters adjusted for spots of this size. The spots are now isolated and the background has zero intensity. There are spots in this image that are obviously not diffraction spots, such as those along the edge of the beam stop. Such spots are removed in

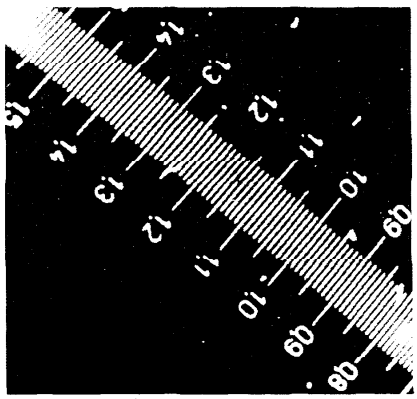

Fig. 11

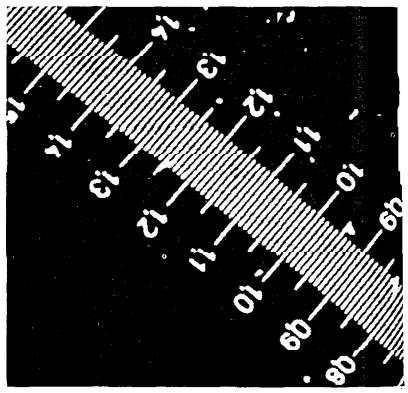

Fig. 12

Fig. 11. Transmission light micrograph of reticle. Threshold set to select lines. All lines are not properly selected due to uneven background illumination.

Fig. 12. Reticle image in Fig. 11 is divided by filtered version of itself (see text) to correct for uneven background illumination. Threshold set to select the lines of the reticle.

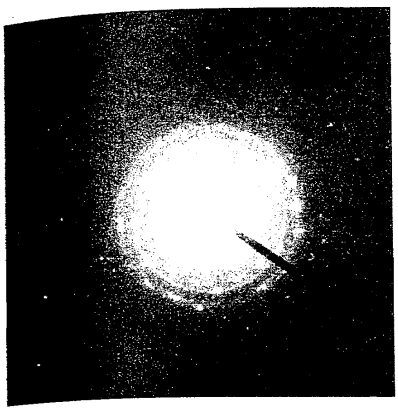

Fig. 13

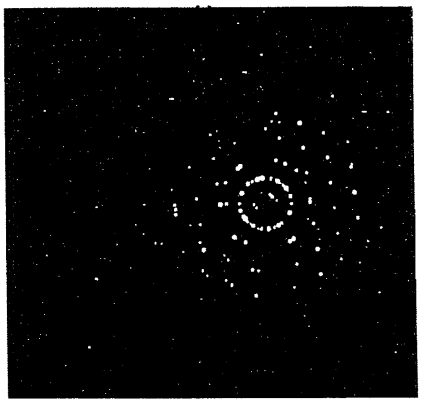

Fig. 14

Fig. 13. Electron diffraction image with spots, diffuse background, rings and shadow of beam stop.

Fig. 14. Result of top hat filter applied to Fig. 13, leaving only spots.

subsequent analysis, along with the single spots from crystallites that do not fall into any regular pattern.

Sometimes the spots are objects to be eliminated rather than selected from an image. Such is the case for Fig. 15, a light micrograph of asbestos fibres and contamination particles that look like spots. The photograph is intended to be made into a template with a calibrated fractional area of the image designated as fibre, thus it was desirable to remove the spots from the photograph. Figure 16 shows the spots alone, after applying the top hat filter with parameters adjusted to fit the majority of spots. Figure 17 shows the original with the spots of Fig. 16 replaced with a background level obtained with the maximum filter. The maximum filter approximates the background because the original is a bright field image. The replacements were done with the MASK command.

(4) *Selection of diffraction rings by averaging.* Occasionally it is better to make a new command than to use existing ones. A case in point is shown by the diffraction pattern in Fig. 18. The top curve is an intensity profile taken along the straight horizontal line through the centre of the pattern. The dim outer rings are often not distinguishable on the intensity profile and when they are, it is due to the profile line fortuitously passing through a diffraction spot on the ring. The bottom curve is an analogous type of plot, except that each value corresponds to an average of all pixels that lie on a circle centered on the beam. The FORTRAN code to do this is simple and faster than using combinations of other commands.

(5) *Setting the threshold automatically to select one phase of an alloy.* Figure 19 is a light micrograph of a polished alloy surface showing two well-defined phases. In order to characterize the sizes and shapes of the dark phase, these dark pixels must first be selected. Uniform setting of the threshold is sufficient to do this, all pixels darker than a certain value belong to the dark phase. The problem is to select the threshold intensity value I_p. Figure 20 shows two representative intensity profiles across the image. The brightness of the two phases is distinctly different—any of a range of values for I_p would separate the two. Figure 21 shows pixels in this range as white and pixels less than that as black. The white pixels correspond to edges of the dark phase regions in this example, and are not insignificant in number even though the edges are sharp. If the edges were not so sharp, the edge pixels would represent an even greater area of the image. In any case, selection of I_p affects many properties of the phase regions, such as area, perimeter and connectivity.

Fig. 15

Fig. 16

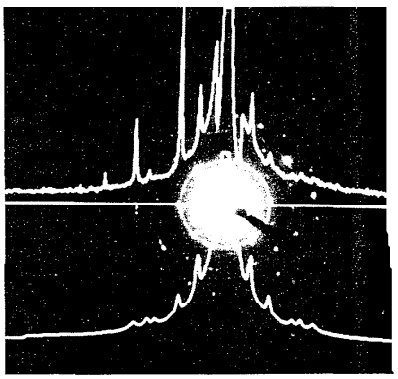

Fig. 17

Fig. 18

Fig. 15. Transmission light micrograph of asbestos fibres and particles.

Fig. 16. Contamination particles isolated from Fig. 15.

Fig. 17. Figure 15 with the contamination particles of Fig. 16 removed (see text).

Fig. 18. Electron diffraction pattern with intensity profile (top) along horizontal line as shown, and rotational average profile (bottom) as described in the text.

Fig. 19. Light micrograph, polished alloy sample. Contrast enhanced.

Fig. 20. Figure 19 with two intensity profiles.

Fig. 21. Figure 19 with pixels of intensities between that of the dark phase and the light phase shown as white.

Fig. 22. Magnitude of gradient of Fig. 19, with contrast enhanced for display.

Fig. 23. Figure 22, but threshold set to show two intensity ranges.

Fig. 24. Figure 20, but with line across intensity profiles showing selected threshold.

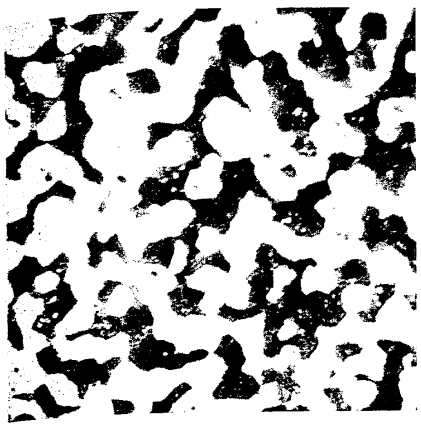

Fig. 19

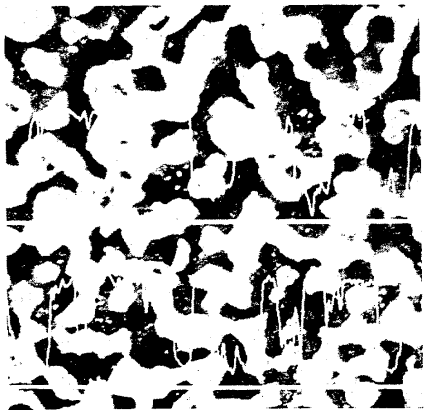

Fig. 20

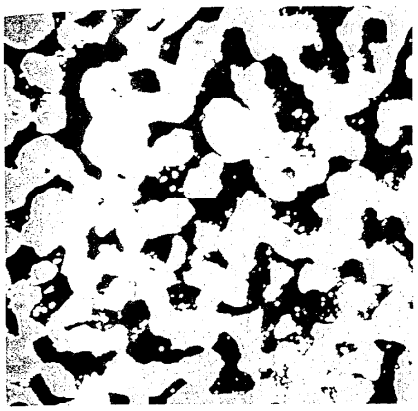

Fig. 21

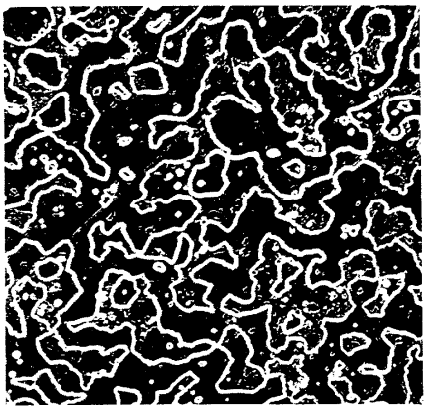

Fig. 22

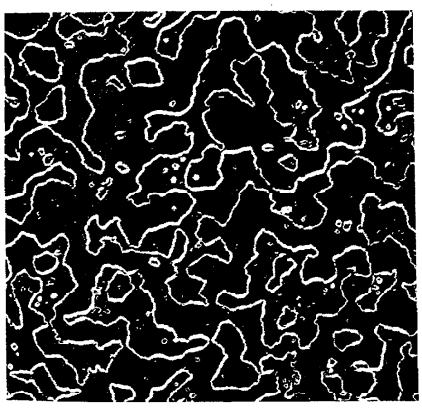

Fig. 23

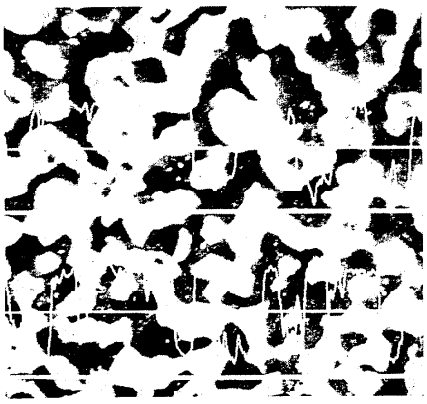

Fig. 24

In this example, since the edges of the phases are sharp, this characteristic alone can b used to automatically select I_p as described below. For a series of images, the value obtained in this way corresponded very closely to thresholds that were independentl chosen by two operators.

To automatically select I_p, edges were enhanced with the GRADA command (a gradie approximation; Rosenfeld, 1982; see Appendix II) as shown in Fig. 22, where the mo abrupt transitions from light to dark in the original image (Fig. 19) appear brighter in th gradient image. Since this image has a range of brightness values, again a threshold must b chosen to select those pixels that correspond to edges. Figure 23 shows the results of tw reasonable thresholds. Such thresholds are selected manually by the appearance of th resulting lines—they should be thin and correspond to the edges in the image. T automatically select a threshold we use the DISTMAP command to measure the thickne of the lines. The threshold is chosen such that, if lowered by one intensity unit, th maximum value of the distance map (half of the maximum thickness not counting th centre pixel, if any) would increase from one to two pixel widths. That is, the threshold is adusted by the computer (see Appendix III, example 4) to get as many edge pixels a possible (by lowering the threshold) with the limitation that the maximum thickness of an edge is only two pixels. The edge pixels are thus chosen from the gradient image. Thes pixels form a mask for selection of the pixels with the same (i,j) from the original image. The average intensity of these (original) pixels is I_p. Figure 24 shows the intensity profile of Figure 20, with the lines superimposed to show I_p.

(6) *Isolating edges to display crystal orientation.* Figure 25 is a transmission ligh micrograph of a glass with crystallite inclusions. To check for any preferred orientations o the crystallites, a polar plot of the histogram of angles from the GRADS command wa made, similar to that of Smith (Smith *et al.*, 1977) except that only edge pixels were included. Plotting the histogram for all image pixels, rather than only edge pixels, obscured the orientation effect for our images.

Figure 26 is the magnitude of the gradient of Fig. 25, where the threshold has been se to select the brightest values, i.e. those that correspond to edges. This image is used t mask or select out the corresponding pixels in Fig. 25, and the polar plot of the histogram of these (edge) angles is shown in Fig. 27. The angle of the polar plot is the angle given by the GRADS command while the radius is proportional to the number of pixels with that angle. The peaks in this plot are perpendicular to the edges of the crystallites, as seen in Fig. 27.

(7) *Line selection using GRADA*, HOUGH* and BLOB† commands.* The Hough transform (Pratt, 1978; Dyer, 1983) maps straight lines into points. If the lines are slightly curved or have thickness, the points become spots. This transform is a useful step because it is easier for algorithms to locate and characterize spots than to locate and characterize lines.

We use the Hough transform in our electron microscope size calibration procedures The task is to measure the line spacing of a well-characterized standard, such as the replica of an optical diffraction grating. Figure 28 shows a transmission electron micrograph o such a replica that has been shaded with a metal to emphasize the lines. The GRADA command enhances the lines, as shown in Fig. 29. Since the HOUGH command works on binary images, the brightest 5% of the pixels are selected to represent the lines as shown in Fig. 30. The HOUGH transform of this image is shown in Fig. 31. A threshold is again set for the transformed image to select the spots. The positions of the spots are obtained with

*See appendix II.

†The term 'blob' is used in the image processing literature (Ferrari *et al.*, 1984) and refers to a connected component or piece of an image (Winston & Horn, 1984).

Fig. 25

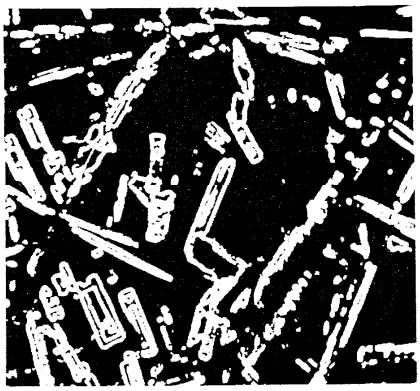

Fig. 26

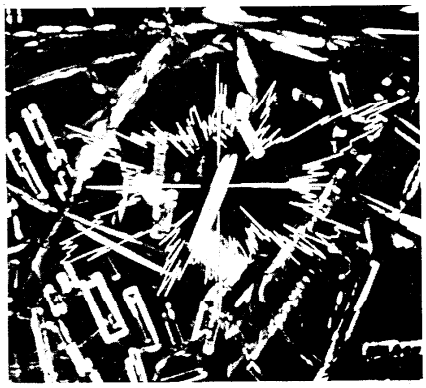

Fig. 27

Fig. 25. Transmission light micrograph of a glass section with embedded crystallites.

Fig. 26. Magnitude of gradient of Fig. 25, with threshold set to select edge pixels (of crystallites).

Fig. 27. Histogram of directions of gradients of edge pixels of Fig. 26, displayed as a polar plot with the number of pixels giving the radius, and the angle of the GRADS command giving the angle. Polar plot is superimposed on Fig. 25 for comparison.

BLOB and LVAL commands. The spot spacings correspond to the spacings of the lines of Fig. 30, and thus of Fig. 28.

(8) *Segmentation by blob selection. Isolation and characterization of individual regions of dark alloy phase.* An image can be segmented into connected or contiguous components or blobs with the BLOB command (see Appendix II). This example uses the image of the two phase alloy in Example 5, where the threshold has already been found that classifies the pixels into the groups corresponding to the two phases. Here, the BLOB command selects the dark pixels and separates them into blobs. Each blob corresponds to one region of the dark phase. Figure 32 shows a mask of one of the blobs, which can be compared directly with the corresponding region in Fig. 19.

Once the blobs are identified, they can be treated as separate objects and have their

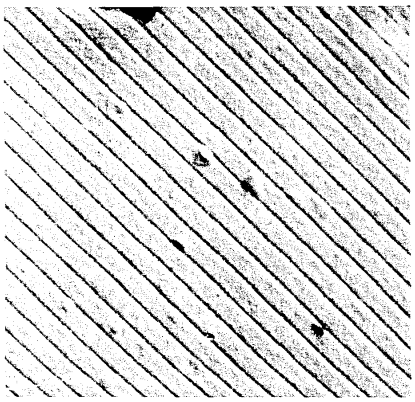

Fig. 28

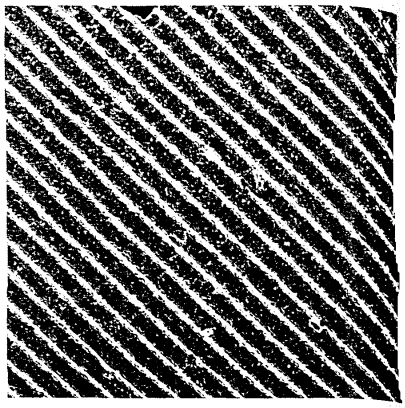

Fig. 29

Fig. 28. Scanning transmission electron micrograph of shaded replica of optical grating.

Fig. 29. Magnitude of gradient of Fig. 28, contrast enhanced.

individual properties analysed. As an example, the blob mask in Fig. 32 is magnified and the part of the original image (Fig. 19) corresponding to it is shown in Fig. 33.

Counting the pixels in the mask gives the area of the blob. This does not yet include the areas of the holes. The holes in the blob can be excluded easily: the blob of the outline (Fig. 34) with the largest bounding rectangle can be selected as the perimeter (Fig. 35), and then the interior can be selected by applying the BLOB command to a binary image of the perimeter. Here, as an alternative, the holes shown in Fig. 36 were selected one by one with the blob command and added back into the mask to give Fig. 37. Either method gives the same result, except that the latter allows each hole to be evaluated separately. For some applications, perhaps only small holes should be filled.

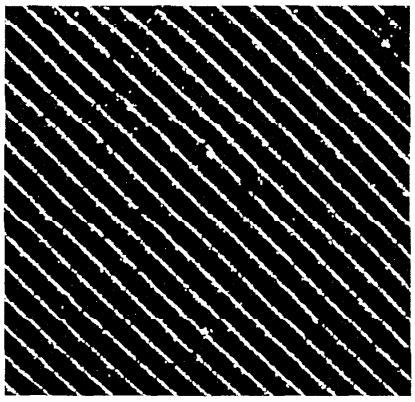

Fig. 30

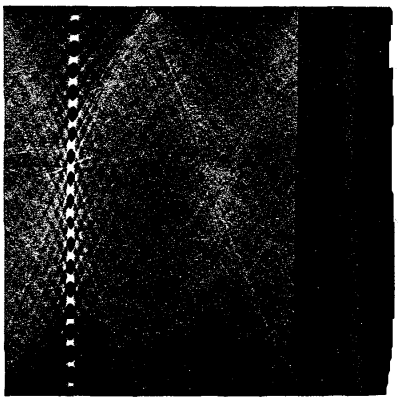

Fig. 31

Fig. 30. Magnitude of gradient, with threshold set to select the 5% brightest pixels.

Fig. 31. HOUGH transform of Fig. 30: Spots correspond to lines in previous three figures.

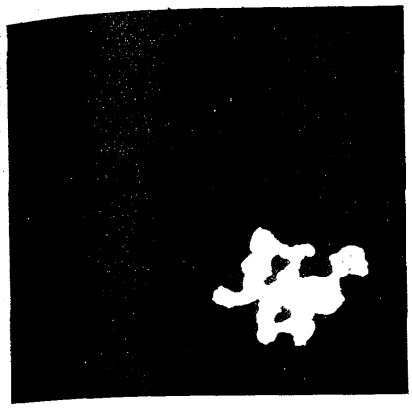

Fig. 32

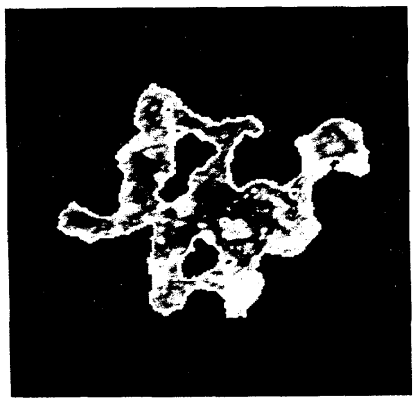

Fig. 33

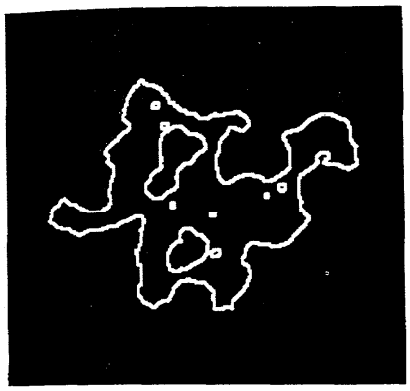

Fig. 34

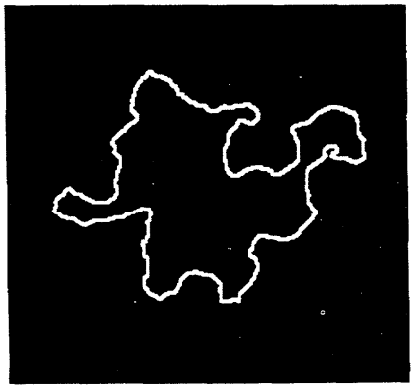

Fig. 35

Fig. 32. Selected blob mask. From Fig. 19, with pixels selected with BLOB command, with the threshold set to value shown in Fig. 24.

Fig. 33. Area of original image (Fig. 19) corresponding to mask of Fig. 32. Magnified ×2.

Fig. 34. Outline of Fig. 32, using OUTLINE command. Magnified ×2.

Fig. 35. Perimeter of area of Fig. 33. Largest connected piece (blob) of Fig. 34 (see text).

Counting all of the pixels of the outline (Fig. 34) gives a rough measure of the perimeter. There are better definitions for perimeter, or the length of a digital curve; however, Rosenfeld has noted that caution should be exercised in using any of them. 'The perimeter of an object often grows exponentially as the digitization becomes finer' (Rosenfeld, 1982). From analytic geometry, the perimeter to area ratio is smallest for a circle. For discrete objects such as images, such is not always the case: 'In digital pictures, depending on how the perimeter is measured, the square of the perimeter divided by the area is smaller for certain octagons than for digitized circles' (Rosenfeld, 1974).

The Euclidian distance map (Danielsson, 1980) is useful for analysing shapes such as that of Fig. 35. The value of the pixel in the distance map is equal to the distance (rounded to an integer) of the nearest background pixel. The distance map, along with the outline for

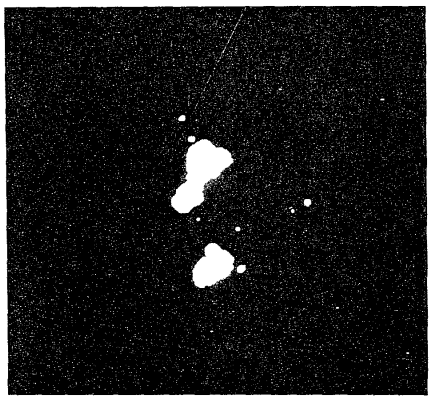

Fig. 36

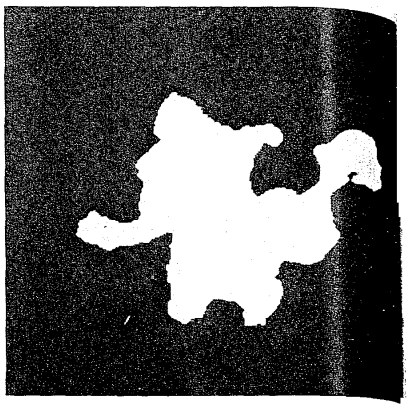

Fig. 37

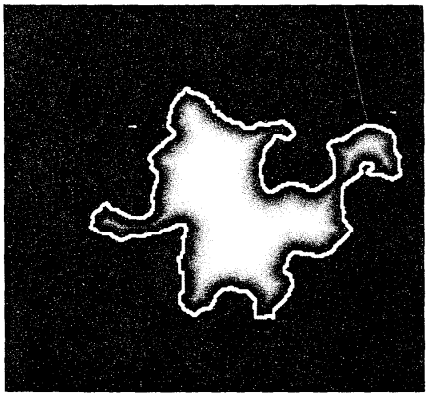

Fig. 38

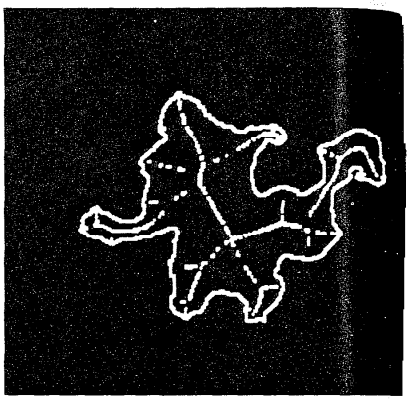

Fig. 39

Fig. 36. Masks of holes in original.

Fig. 37. Original mask with holes filled in.

Fig. 38. Distance map of Fig. 37 (see text).

Fig. 39. Ridge points of Fig. 38, shown with outline for reference. These points approximate skeleton original shape.

Fig. 40. Ion microscope image where one spot was selected with an aperture on the instrument.

Fig. 41. Least significant byte of original two byte image, Fig. 40. Original now shown with extreme cont enhancement (with lots of wrap-around). Extent of instrument imaging aperture is shown as white surrounding the spot.

Fig. 42. Original ion microscope image, contrast enhanced to show dark skirts, in an attempt to show ex of blooming of spots.

Fig. 43. Direction of gradient of image in Fig. 42. Directions are displayed as intensities as in Fig. 44. E of blooming of spots is easily seen.

Fig. 44. Polar plot of special function of direction (as intensity or radius) vs. angle that is useful displaying the direction of the GRADS command, superimposed on the direction of gradient image f conc.

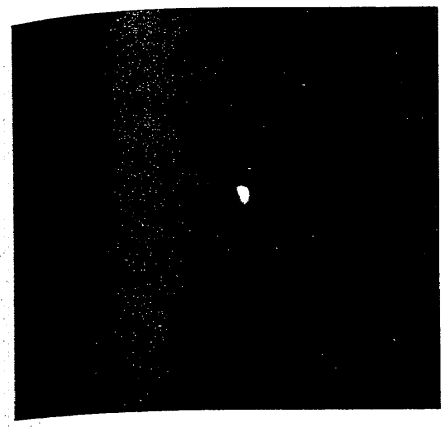

Fig. 40

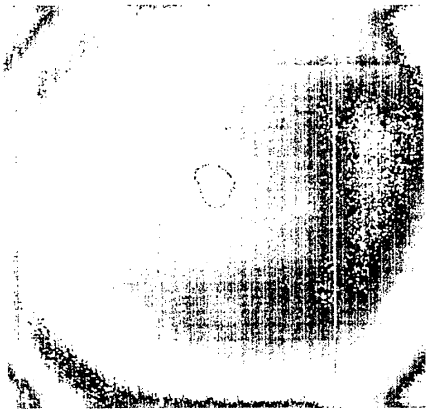

Fig. 41

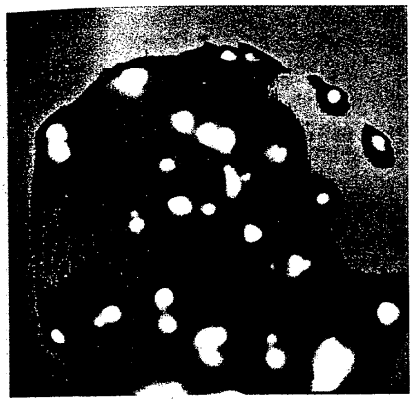

Fig. 42

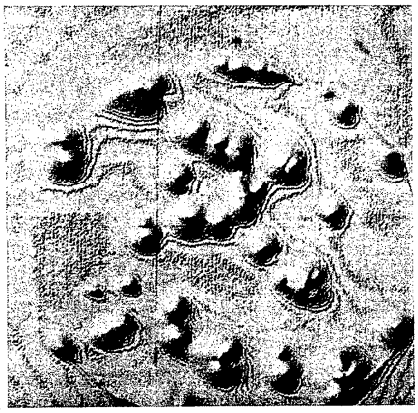

Fig. 43

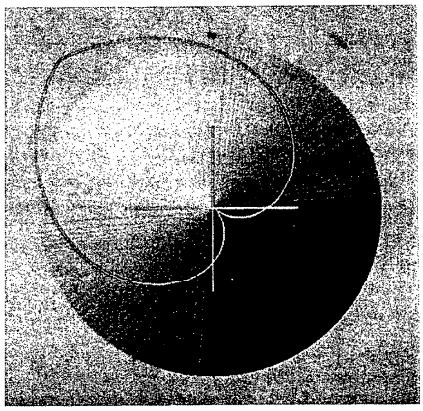

Fig. 44

clarity, is shown in Fig. 38. Note that the brightest points are those that are farthest from any point on the outline. Examination of the distance map helps to characterize shape. Pixels approximating the skeleton of the blob are shown in Fig. 39. The values of the distance map for these pixels indicates how much of the blob consists of long thin arms or fat rounded parts. The skeleton approximation was determined from the distance map as pixels that are on a one or two pixel wide local maximum, horizontally or vertically. Since this skeleton approximation is still under evaluation, it is not given here as part of the LISPIX system. Skeletonization for image analysis of other types of metallurgical samples is discussed in detail for a hexagonal array (Lantuejoul, 1980), and a variety of methods for generating skeletons on square arrays appear in the literature (Pavlidis, 1982; Russ, 1984; Salario & Siy, 1984; Zhang & Suen, 1984; Arcelli & Di Baja, 1985).

(9) *Detection of aperture in ion microscopic image.* An aperture can be used in the ion microscope to select a portion of an image for intensity measurements. It was possible to detect the extent of the aperture by displaying only the least significant byte of each pixel (this amounts to extreme brightness contrast enhancement). Figure 40 shows the original image of the spot where the aperture is not discernible. Figure 41, the least significant byte of Fig. 40, shows the disc-shaped aperture as the white area surrounding the spot.

(10) *Determination of extent of blooming of spots in ion microscope images.* Sometimes the images of particles in the ion microscope show blooming due to various instrumental effects. A display of the direction of the gradient of the image is useful for showing the extent of the blooming. Figure 42 shows an original ion microscope image of some particles. Figure 43 shows the direction of the gradient, displayed as explained below. Note that the extent of the spots is considerably larger than is evident from the original image.

Values of the direction of the gradient can vary from 0° to 360°. For an intuitive image, we use a function of the direction that has no abrupt changes in value for small changes in direction, and that has equal brightness values for deviations for both clockwise or counter-clockwise deviations from the direction of maximum intensity (Appendix III, Example 3). Figure 44 shows the function used in this work, which displays peaks, such as the cone on which the plot is superimposed, as if they were illuminated from the upper left.

REFERENCES

Arcelli, C. & Di Baja, G.S. (1985) A width-independent fast thinning algorithm. *IEEE Trans.* PAMI-7, 463–474.

Bright, D.S. & Steel, E.B. (1987) Two dimensional top hat filter for extracting spots and spheres from digital images. *J. Microsc.* **146,** 191–200.

Danielsson, P.E. (1980) Euclidian distance mapping. *Computer Graphics and Image Processing,* **14,** 227–248.

Dyer, C.R. (1983) Gauge inspection using Hough transforms. *IEEE Trans. Pattern Anal., Mach. Intel.* PAMI-5 (6), 621–623.

Ferrari, L., Sankar, P.V. & Sklansky, J. (1984) Minimal rectangular partitions of digitized blobs. *Computer Vision, Graphics and Image Processing,* **28,** 58–71.

Goldstein, H. (1950) *Classical Mechanics.* Addison Wesley, Reading, Mass.

Goldstein, J.I., Newbury, D.E., Echlin, P., Joy, D.C., Fiori, C. & Lifshin, E. (1981) *Scanning Electron Microscopy and X-ray Microanalysis.* Plenum Press, New York.

Heinrich, K.F.J. (1981) *Electron Beam X-ray Microanalysis.* Van Nostrand Reinhold, New York.

Hren, J.J., Goldstein, J.I. & Joy, D.C. (1979) *Introduction to Analytical Electron Microscopy.* Plenum Press, New York.

Huang, S.T., Yang, G.J. & Tang, G.Y. (1979) A fast two-dimensional median filtering algorithm. *IEEE Trans. on Acoustics, Speech and Signal Processing,* ASSP-27, 13–18.

Kirsch, R.A. (1971) Computer determination of the constituent structure of biological images. *Comput. Biomed. Res.* **4,** 315–328.

Knuth, D.E. (1973) *The Art of Computer Programming, Vol.* 3, *Sorting and Searching.* Addison Wesley, Reading, Mass.

Landy, M.S., Cohen, Y. & Sperling, G. (1984) HIPS: a UNIX-based image processing system. *Computer Vision, Graphics and Image Processing,* **25,** 331–347.

Lantuejoul, C. (1980) Skeletonization in quantitative metallography. In: *Issues in Digital Image Processing* (ed. by J. R. Haralick and J. C. Simon), pp. 107–135. Sijthoff and Noordhoff, Germantown, Md.

Marinenko, R.B., Myklebust, R.L., Bright, D.S. & Newbury, D.E. (1987) Digital X-ray compositional

mapping with 'standard map' corrections for wavelength dispersive spectrometer defocusing. *J. Microsc.* 145, 207–223.
Meinzer, H.P. & Engelmann, U. (1984) The interpreter PIC: a tool in the field of image processing. *Med. Inform.* 9 (2), 93–102.
Natrella, M.G. (1963) *Experimental Statistics*. National Bureau of Standards Handbook 91, U.S. Government Printing Office, Washington, D.C.
Orser, D.J., Rhodes, I.I. & Meninger, A.H. (1972) The LISP image processing facility on the PDP-10. NBS Report 10850, 30 June 1972, Department of Commerce, National Bureau of Standards, Gaithersburg, Md.
Pavlidis, T. (1982) *Algorithms for Graphics and Image Processing*. Computer Science Press, Rockville, Md., pp. 195–214.
Pratt, W.K. (1978) *Digital Image Processing*. Wiley, New York.
Rosenfeld, A. (1974) Compact figures in digital pictures. *IEEE Trans.* SMC-4, 211–223.
Rosenfeld, A. & Kak, A.C. (1982) *Digital Picture Processing*, 2nd edn, Vol. 2. Academic Press, New York.
Rosenfeld, A. (1984) Image analysis. In: *Digital Image Processing Techniques* (ed. by M. P. Ekstrom), p. 257. Academic Press, Orlando, Fla.
Russ, J.C. (1984) Implementing a new skeletonizing method. *J. Microsc.* **136**, RP7–RP8.
Russ, J.C. (1986) Personal communication.
Salari, E. & Siy, P. (1984) The ridge-seeking method for obtaining the skeleton of digital images. *IEEE Trans.* SMC-14, 524–528.
Smith, K.C.A., Unitt, B.M., Holburn, D.M. & Tee, W.J. (1977) Gradient image processing using an on-line digital computer. *Scanning Electron Microscopy*/1977, Vol. I. Scanning Electron Microscopy, Inc., A.M.F. O'Hare, Chicago, Ill.
Tamura, H., Sakana, S., Tomita, F., Yokoya, N., Kaneko, M. & Sakaue, K. (1982) Design and implementation of SPIDER—a transportable image processing software package. *Computer Vision, Graphics and Image Processing*, **223**, 273–294.
Winston, P.H. & Horn, B.K.P. (1984) *LISP*, 2nd edn. Addison Wesley, Reading, Mass.
Zhang, T.Y. & Suen, C.Y. (1984) A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, **27**, 236–239.

Appendix I. Command Syntax and Overview of LISPIX Commands

Command syntax

The following command syntax is used because it is simple to implement. It has remained unchanged even while the system has grown in size due to adding new commands. LISPIX commands consist of one line with:

(a) a command name
(b) zero or more parameters
(c) an optional bounding rectangle.

All parts of the command are separated by spaces. The command interpreter divides the command into its parts, keeping them in order. The command name determines the operation to be done. This is implemented by having the first part determine which subroutine to call to actually perform the command. The subroutines corresponding to each command get all of the command parts, and interpret them according to their order as defined below.

The parameters can be image arrays, pixel values, pixel locations, scale factors or special constants. Image arrays are referred to as letters (or pairs of letters for more than 26 image arrays) to avoid confusion with other parameters.

The bounding rectangle is the area of the image on which the command will be performed. The bounding rectangle is a list of four numbers ($x1$ $y1$ $x2$ $y2$) giving the coordinates of the lower left and upper right corner of the rectangle. If it is omitted, a default rectangle is used. If the default has not been specified with the BDR command, it is assumed to be the entire image.

Examples of commands are:

```
BDR 100 100 200 200
IADD a b c
THRESH e d 58 255 10 10 20 20
```

The BDR example sets the default bounding rectangle. The bounding rectangle is th image area affected by subsequent commands. In this example it is the 101×101 pix square with the lower left-hand corner at $(x,y)=(100,100)$ and the upper right-hand corn at $(x,y)=(200,200)$. The IADD example adds the image in array a to that in b, pixel b pixel, and stores the result in array c. (The letters designating arrays are underlined in th text for clarity.) The area operated on is whatever the default bounding rectangle is at th time, which can be anything from a single pixel to an entire 512×512 pixel image. The las example sets the thresholds for image e to form a binary image in d. Pixels in d are 1 whe pixels in a are between 58 and 255, inclusive, and are zero otherwise. In this case, brigh objects are being selected, and 255 is the maximum image intensity. If dark objects were t be chosen and zero were the minimum image intensity, the thresholds would be 0 and 5 The THRESH command in this example works on the 11×11 pixel area with the lowe left-hand corner at pixel 10,10. Pixels in array d that are outside this area are lef unchanged.

The LISPIX commands can be grouped by function: input/output, graphics, processin and analysis.

Input/output commands. These commands are used for image storage. They do no pertain directly to the analysis algorithms, and will not be discussed here, except for th following details which proved useful:

A two byte integer is not needed to store each pixel when the range of pixel values i limited, as is often the case. Also, only portions (bounding rectangles) of the arrays ar sometimes stored or displayed. A small header is thus recorded with each image that give the type of image, the range of pixel values and the bounding rectangle of the image portion. This simple scheme has remained satisfactory for several years. Using this header analysis algorithms may read the size of an input image. Sometimes other image headers ar used for special purposes, such as recording instrument settings for later use by LISP functions. The additional information in these headers is never used by LISPIX for readin and writing images.

A common format for recording 512×512×1 byte images is in 512 records of 512 byte each. There is no information in the file except pixel information. Since this is such common type of format, there are special LISPIX commands to read and write the images. They give compatibility with other image processing systems and save disc space.

Graphics and display commands. These commands are for graphics, image display and annotation. Although they represent considerable work and are used often, they are no discussed here, except again for the following details which proved useful:

Image and graphics display devices often differ between sites, whereas FORTRAN and LISP are available on a variety of virtual memory machines. The system therefore does as much as possible in the host computer, and uses a minimum number of display and graphic commands. This facilitates transfer of LISPIX to other systems with different display devices.

Useful graphic commands are those that write various simple geometric objects such as lines, arcs, sectors and alphanumeric characters. These are used for outlining objects, for displaying intensity profiles and for annotating images. Since we record many images on film, annotation of each image including subject, source, time and date are essential.

We routinely have an image displayed in one colour (green works best) and graphics displayed in another colour. Since it is difficult to publish in colour, we use a command that takes the image in one array, the graphics in a second array and forms a combination. Figures 27 and 44 were made using this command. The image is left unchanged except where covered by the graphics, which is white with a one-pixel-thick black border.

Processing. Processing commands have an image as a result. The simplest group do uniform pixel operations, that is, they operate on each pixel independently of the other pixels. They include the threshold setting, arithmetic and logical operations.

(1) Single pixel LISPIX commands.

Command	Description
THRESH	set the threshold—gives a binary image
ISUB	integer subtraction
IDIV	integer division
SDIV	scaled integer division
SCALE	scale image to have a given maximum value
MASK	transfer image pixel if mask pixel is within threshold

The THRESH command (explained in the example above) is often used to segment an image into object pixels and background pixels. Even presuming a uniform constant background, this is a very often used but error-prone operation. Objects are often different in intensity from the background, but the edges are usually thick, and different thresholds will select different proportions of the edge. Sometimes an algorithm can be used to reliably select the threshold (Example 5) but other times the threshold must be chosen by the operator who decides what is object and what is background.

ISUB is used to remove background-level intensity where appropriate. This is used for images from the ion microscope, where there is some additive background signal from both the channel plate detector and the television camera.

SCALE is used to change the dynamic range of an image for display, and for multiplication and division since integer arithmetic is used for these operations.

Division is appropriate for transmission light microscope images that need correction for uneven illumination (see Example 2). The commands that perform division are IDIV and SDIV. Since division by zero is not defined and images, especially at the boundaries, often have pixels with value zero, both commands actually replace the zero values of the divisor pixels with unity. The IDIV command is often awkward to use, requiring proper scaling of the numerator image so that the quotient image will have sufficient dynamic range. The SDIV takes care of that by truncating the quotient values to integers only after they have been scaled to a maximum value of 255, the maximum value that can be displayed. Division and multiplication are appropriate in special cases for scaling images (Appendix III, Example 1).

The MASK command extracts objects in an image by copying only the image pixels that correspond to mask pixels. In binary images, mask pixels have unity value whereas background pixels have zero value. Masks usually correspond to objects in shape and are usually obtained by setting the threshold of an image or using the BLOB command.

(2) Other uniform pixel commands. These LISPIX commands are used frequently, but not as often as the previous ones:

Command	Description
IMUL	integer multiplication
FILL	fill bounding rectangle with pixels of given value
IADD	integer add
IABS	absolute value
IMAX	maximum value
IMIN	minimum value
SQRT	square root (truncated to integer value)
LOG	logarithm (truncated to integer value)

The IMUL is used occasionally, for example, to enhance the diffusion region between two adjacent phases. When each of two elements in two registered electron probe images is abundant in one phase but not in the other, the diffusion region with both elements will have higher values in the product image than either phase.

The IADD command has several uses. The most common one is to collect separate parts or areas of an image together by successively adding them to an 'accumulator' image array. Examples of this are: (a) collecting all of the blobs (or blob masks) together, (b)

filling in holes in a mask (Example 8) and (c) filling in holes in images with a synthesized background (Example 3). There is a copy command that might seem to be the logical choice for these operations, however the copy command moves the entire bounding rectangle, blank or zero areas and all. Where objects have overlapping bounding rectangles, use of copy results in partial erasure of one of the objects or masks.

Some of the images come from the image processor to the LISPIX system in the form of two one-byte images. IADD is also used to add the upper and lower bytes together (using ishift appropriately) to form a two-byte image.

The IABS command is used for displaying deviations where only the magnitude of the deviation is important. Otherwise, when scaled for display, the zero pixels are assigned a grey value, the most negative pixels are displayed as black and the most positive ones as white. After using IABS, the zero deviation pixels are black and the ones with large negative or positive deviations are white. This is also used to display the direction image from a gradient calculation, where the values range from 0° to 360°. To avoid having an abrupt intensity change for the small change in angle at 0°, the angles are shifted to range from −180° to 180°, and the absolute values are displayed (see Fig. 44 and also Appendix III, Example 3, line 10).

The IMAX and IMIN commands have been used to get a background intensity image from two or more images where the objects do not overlap so that a background level is available for each pixel in at least one of the images.

The SQRT and LOG commands are sometimes used to compress certain intensity ranges for display, when manipulation of the display intensity look-up tables will not suffice. This is sometimes the case because the dynamic range of the display device is 0 to 255, but the dynamic range of the image arrays is −32,768 to 32,767.

(3) Movement and distortion. All of these commands move a pixel to a different position, rather than change its value.

Command	Description
MAG_BILIN	*x* and *y* stretch or shrink*
MAG_NN	*x* and *y* stretch or shrink†
ROTATE_BILIN	rotate about a given centre point*
ROTATE_NN	rotate about a given centre point†
SHRINK	decrease image size
TRANS	translation

The MAG_BILIN and MAG_NN commands will stretch (or shrink) an image in the horizontal and vertical directions from a given centre. This has been used where the aspect ratio of images has been changed in transferring from one system to another, or where objects with slightly different aspect ratios must be overlapped.

The ROTATE_BILIN and ROTATE_NN commands will rotate an image by a given angle about a given centre. This has been used to align images.

The SHRINK command reduces the resolution and size of an image by averaging an $n \times n$ pixel square and putting the rounded mean value in one pixel. This has been used to cut processing time on an image, and to increase the effective range of some of the gradient commands. This can be appropriate for processing objects that are fuzzy in the original image, that is when there are more pixels per object than is justified by the instrumental resolution.

The TRANS command moves a portion of an image specified by the bounding rectangle to a new location. This is used for aligning images and for displaying several small images at one time.

(4) Window commands. These commands use relations between the neighbouring pixels within a window centered on a given pixel in an input image. The result of arithmetic

*Bilinear interpolation.

†Nearest neighbour interpolation.

operations on the pixels in this window are placed in the output image in the corresponding position to the given pixel in the input image. The OUTLINE command uses only a 3×3 pixel window. The other commands can use larger square windows specified by a width parameter, w, for a window $2w+1$ pixels on an edge. When using these windows a w width pixel border inside the current bounding rectangle of the resulting image is undefined. Undefined pixels are left unchanged.

Command	Description
MEAN_FILTER	average value of window
MEDIAN_FILTER	median of window
RANK_FILTER	min, max or selected rank of window
TOP_HAT	specialized top hat, spot isolating filter
CONVOLVE	convolution
OUTLINE	outline of masks

Smoothing, or low pass filtering, is often done with the MEAN_FILTER command (Rosenfeld & Kak, 1982) and is a special case of convolving. We use it fairly infrequently.

The MEDIAN_FILTER has been particularly useful in eliminating scratches from X-ray maps of standards. The standard maps are used to correct for changes in instrument sensitivity over the field of view, and scratches and small surface defects of the standard sample deteriorate the results. We use a median filter with w roughly equal to the width in pixels of the widest scratch to be eliminated (see Example 1). We also prefer the median filter over the mean filter because the median filter is much less sensitive to single pixel noise (salt and pepper noise) and preserves edges better (Pratt, 1978).

The RANK_FILTER command, a generalized version of the MEDIAN_FILTER command, is useful as a maximum or minimum filter for reconstructing a background image (see Example 2).

TOP_HAT, the top hat filter, is used for isolating spots. The spots have been either items for selection, such as electron diffraction spots, or items for removal, such as dust spots from images of fibres (Example 3). The top hat filter is an example of a gated template (Rosenfeld, 1984, p. 266) that has the form of a disc surrounded by a ring. Pixels from the original image are transferred to the filtered image only if the largest pixel value in the disc exceeds the largest in the ring by a given tolerance.

The CONVOLVE command can have a variety of results depending on what the kernel values are (see Pratt, 1978, p. 319). The convolution operations are mentioned here because they are commonly used elsewhere. We do not use them, primarily because they seem to blur more than they reduce noise. Convolutions are often used to take Laplacians and directional derivatives or gradients (Pratt, 1978). These operations are used to advantage for image enhancement, but for our analysis procedures, we prefer non-linear filters and specialized gradient commands.

The OUTLINE command gives edges of masks (binary images). The edges themselves are masks, and are contiguous (four-connected). The four-connected property means that the BLOB command will give one blob for each edge.

(5) Gradients or edge enhancers.

Command	Description
GRADA	fast approximation of magnitude of gradient
GRADS	magnitude and direction or gradient

These gradient commands calculate the changes in intensity between neighbouring pixels. Like the window commands above, the gradient commands also look at values of neighbouring pixels, but only the neighbours directly above, below and to both sides.

The magnitude images of both GRADA (Rosenfeld & Kak, 1982) and GRADS (Smith *et al.*, 1977) look much the same, although they are calculated differently (Appendix II). They are both useful in enhancing edges of objects, as in Figs. 22 and 29.

The direction image of the GRADS command can be used to detect orientation effects by tallying the orientations of edge pixels (see Example 6) and is sometimes useful as a processing operation for enhancing regions of low intensity or for showing the total extent of objects with fuzzy edges (Fig. 43).

(6) Transforms. The following transforms are used on binary images, where the division between object and background pixels has already been made. The results are grey scale (multi-valued) images.

Command	Description
HOUGH	*Hough transform*
DISTMAP	Euclidian distance map

The Hough transform maps lines into points, or, for our images, it maps long, thin objects into blobs or dots. The resulting image from this transform is used to detect these objects in several applications and measure their spacings (see Example 7). The transform is discussed in the image processing literature (Pratt, 1978; Rosenfeld & Kak, 1982; Dyer, 1983).

The DISTMAP command (Euclidian distance map) gives the shortest distance from an object pixel to the background. The values are all zero outside the object. This map is useful for measurements on complex shapes (see Example 8).

Analysis command. Segmentation with the BLOB command.

Command	Description
BLOB	segments image into blobs

The BLOB command divides a binary image into groups of mutually contiguous pixels, or blobs, and background (Kirsch, 1971; Winston & Horn, 1984). Each pixel in a blob is adjacent on at least one of the four sides to another pixel of the same blob. Under the right conditions, these blobs correspond to objects of interest in the image. The BLOB command is often not executed with an original image but with one that has been transformed with the GRADA, TOP_HAT or HOUGH commands. For example, the BLOB command was used on Fig. 31 to extract the spot positions that correspond with the line spacings of Fig. 28. Several intermediate images were involved in producing Fig. 31 from Fig. 28.

This segmentation using BLOB reduces the image to a set of objects rather than a much larger set of object pixels. The blobs are often the items used in the LISP image analysis algorithms.

Information summary commands

Information about an image is passed to the LISP algorithms with these commands:

Command	Information returned to Lisp
BAREAS	list of all blob areas, sorted largest first
HIST	histogram
LIMITS	minimum and maximum value within a bounding rectangle
LVAL BLIST	a list of blob properties (area, maximum value, etc.)
LSTATUS	number of blobs, image directory or default bounding rectangle
SLINE	values along an intensity profile

The most often used commands give information about blobs, and are used immediately after the BLOB command is used: BAREAS, LVAL and LSTATUS. BAREAS (blob areas) returns a list of all of the blob areas, largest first. LSTATUS BLIST returns a count of all blobs greater than the given area. LVAL BLIST returns a list of information about a specific blob. Both LSTATUS and LVAL have other uses—the BLIST key word specifies information kept in the blob list (Fig. 1).

This is an example of the information on a single blob (number 4) that is retrieved from

the blob list with the LVAL BLIST command, typed as LVAL BLIST 4:

((area 5929) (s_num 1) (bdr (4 4 80 80)) (thresh (10 255)) (l_x_v (30 30)) (x_v 50) (l_n_v (4 4)) (n_v 10) (av_v 12.97) (sigma 10.49) (c_mass (41.54 41.54)) (unique t))

The serial number or identification number, s_num, is arbitrary but unique for each blob. Every pixel of the blob is set to this value (see Appendix II) for identification and for use with the MASK command to extract a single blob from an image. The area is the number of pixels in the blob. The bounding rectangle is the blob location given as the lower left and upper right-hand corners of the bounding rectanagle of the blob. Thresh is the BLOB command intensity range. L_x_v, x_v, l_n_v and n_v are the locations and values of the maximum and minimum pixels of the blob. The rounded mean value and standard deviation (the square root of the variance, see Natrella, 1963) of the blob pixels are given as av_v and sigma. C_mass is the average pixel location of an object, weighted by pixel intensity and thus is a centre of 'mass'. The centroid, which is the average pixel location with equal pixel weights, can be obtained from the c_mass of the mask of the object, since all of the pixels have equal weight.) The unique parameter is 't' if there are no other blobs in the same bounding rectangle for the same thresholds. If this condition holds, then the blob can be reconstructed with the faster THRESH command, rather than with the BLOB command. This is useful because it is usually more economical to reconstruct blob masks when necessary rather than to store them.

HIST returns a histogram. We use this occasionally to determine the median values of image areas, to count pixels within a given intensity range, or to quickly find an intensity range that includes, say, the 5% brightest pixels.

LIMITS returns the maximum and minimum values of an image.

SLINE returns a list of pixel values along a straight line. This is usually used for plotting intensity profiles.

Special purpose commands

Sometimes it is appropriate to write a special purpose command rather than use other commands within LISP functions. The rings command is an example of this (see Example 4), where the FORTRAN code is particularly simple (which is not the case for the FORTRAN equivalent of most LISP functions) and much faster than using combinations of existing commands. The system is organized so that new commands can be easily inserted into the system.

APPENDIX II. DEFINITIONS OF LISPIX COMMANDS

Introduction

This appendix gives definitions of the commands that operate on arrays. Other commands are described elsewhere in the text.

Arrays in the example command statements are referred to by the letters *a*, *b* and *c*, and in the text by a, b and c for clarity. Also only for clarity, command names are in upper case—LISPIX uses either upper or lower case.

Unless specified, the operation is to be applied for all *i* (columns) and *j* (rows) in the bounding rectangle. For example, if full 512×512 images are used, then

$$i=1\ldots512$$
$$j=1\ldots512$$

and every pixel is included. For clarity, the bounding rectangle is not included in the definition—the default bounding rectangle applies.

For window operations, the boundary of the image is commonly excluded. This is the case for all commands that have a width parameter, *w*, in the given definition, where the excluded boundary is *w* pixels wide. For example, consider a full 512×512 pixel image in

the input array a. The range of the indices for the operation would then be:

$$i=1+w \ldots 512-w$$
$$j=1+w \ldots 512-w$$

The first w and last w columns and rows of pixels are excluded from the operation. The corresponding pixels in the output matrix are left unchanged.

The letter 'w' is replaced with an integer when using the system, and has been used here only for clarity. Otherwise, all of the example command statements can be used as they appear in the text. The '##:' prompt is reproduced in front of these statements to make them stand out.

Unless specified, an array may be used more than once, for example:

##: IADD a a a
or
##: IADD f c f
is allowed, as is
##: IADD a b c

The first command will double the values in a, the second command will add the values in c to f, while the last command replaces the values in c with the sums of the respective values in a and b.

Definitions

The definitions are given in FORTRAN like notation. The commands are implemented in FORTRAN.

##: BLOB a b 100 200

Every $b(i,j)$ is a given the identification number of the blob for which it is a member, if $a(i,j)$ is within the threshold range, i.e. if $100 \leqslant a(i,j) \leqslant 200$. Otherwise, $b(i,j)$ is set to zero.

A blob is a set of pixels that are four-neighbour continuous, that is every pixel of the blob is adjacent on the top, bottom, left or right, to at least one other pixel of the blob.

Although the eight-neighbourhood (the four-neighbourhood plus the four corner pixels) is often used (Winston & Horn, 1984), LISPIX uses the four neighbourhood to avoid ambiguous conditions that can occur with the eight-neighbourhood: a diagonal line with pixels touching only at the corners would be an object, and two such objects can cross 'x' fashion yet both remain connected. Normally, one of these stringy objects would have to break as it passes under the other. A four-connected object implies an eight-connected background, and vice versa. If background areas such as individual holes in objects are if interest, the blob command can momentarily treat the holes as objects (and the object as background) so that the holds are four-connected. This is rarely an issue in the laboratory because we avoid experimental conditions resulting in objects only one pixel thick.

The blob algorithm involves scanning the image once to label the pixels while keeping track of equivalent labels, and the scanning the output image again to change labels where necessary so that each label is unique to a blob (Winston & Horn, 1984, p. 163).

##: CONVOLVE a b

$$b(i,j)=\sum a(i+l,j+m)*k(l,m)$$
$$l=-w,w$$
$$m=-w,w$$

Exclude a border, width w, of pixels in b inside the bounding rectangle.
k is the convolution kernel, written by LISP from a library of kernels, or generateed by a LISP function as needed. w is determined automatically from the size of k. For discussions of convolutions of this type, see Pratt (1978, p. 319) and Rosenfeld & Kak (1982, p. 24).

##: DISTAMP a b 30 55
Let object pixels in a be object pixels if

$$30 \leqslant a(i,j) \leqslant 55$$

and background pixels otherwise. Then $b(i,j)$ is given the integer value of the Euclidian distance (straight line distance) from the centre of $a(i,j)$ to the centre of the nearest background pixel in a. The algorithm, 8SED, is given by Danielsson (1980).

##: FILL a 33

$$a(i,j)=33$$

The value assigned to the pixels can be an integer value from −32,768 to 32,767.

##: GRADA a b
GRADA and GRADS use the four-neighbourhood. The following notation helps clarify the definitions of these commands.

$$\begin{array}{ccc} & n2 & \\ n3 & p & 1 \\ & n4 & \end{array}$$

where p is the centre pixel intensity, $a(i,j)$, and $n1$, $n2$, $n3$, and $n4$ are the neighbouring pixel intensities:

$$a(i+1,j),\ a(i,j+1),\ a(i-1,j),\ a(i,j-1).$$

GRADA, from Rosenfeld & Kak (1982), is defined as:

$$b(i,j)=\max(|n1-n3|,|n2-n4|)$$

where $|x|$ is the absolute value of x and where $\max(x,y)$ returns either x or y, whichever is greater. This is a fast approximation to the magnitude of the gradient that we use most often.

##: GRADS a b c
From Smith *et al.* (1977). b receives the magnitude values and c the direction values in degrees:

$$b(i,j)=\text{int}(\text{sqrt}((n1-n3)**2+(n2-n4)**2))$$

$$c(i,j)=\text{int}(\text{atand2d}(n2-n4),(n1-n3))$$

and, if $c(i,j)=0$, then $c(i,j)=360$

where $\text{atan2d}(y,x)=\arctan(y/x)$ where the signs of the numerator and denominator are taken into account to give angles over a full circle. Zero degree values have been replaced with the equivalent 360°, to distinguish them from zero value background pixels. Int(x) is the truncated integer value of x, and sqrt(x) is the square root of x.

##: HIST a
Returns a list giving the value and the number of pixels having that value. For example, if the bounding rectangle is 1 1 10 10 and there are twenty pixels with intensity 100, thirty pixels with intensity 150, and the rest with intensity zero, then HIST returns the list:

((0 50) (100 20) (150 30))

as the histogram.

##: HOUGH a b

This command puts in b the Hough transform (a specialized histogram) of the binary image in a. The transform has the effect of mapping a point in a into a sinusoidal curve in b but the cumulative effect is for straight line segments in a to be mapped into points or dots in b.

The algorithm and coordinate systems used are very similar to those by Dyer (1983) but have been optimized for our 512×512 byte images. For this transform only, the origin of the image in a, is at the centre, C, of the array, i.e. at a(256,256). Consider a line segment L in a, and the perpendicular line P from L or its extensions to the centre. The length of P is the distance from L to C and is represented as the ordinate in array b. This length can be negative if the intersection of L and P is below C—values range from −225 pixels to 256 pixels in steps of one pixel. The angle P makes with the abscissa is represented as the abscissa in b—and ranges from 0° to 179·5° in steps of 0·5°.

The bounding rectangle applies to a as needed, but is usually 1 1 512 512. Due to the change in coordinate systems, the bounding rectangle does not apply at all to b.

For every non-zero pixel in a, a pixel in every column of b is incremented as follows:
For each non-zero $a(v,w)$, for $i=0, 0{\cdot}5, 1, \ldots 179{\cdot}5$:
increment $b(i,j)$ if $1 \leqslant j \leqslant 512$, where

$$j=256+\text{int}((v-256)*\cos(i/2)+(w-256)*\sin(i/2))$$

Illustrated examples of this transform are given in the literature (Pratt, 1978, p. 524; Dyer, 1983).

##: IABS a b

$$b(i,j)=\text{abs}(a(i,j))$$

##: IADD a b c

$$c(i,j)=a(i,j)+b(i,j)$$

##: IDIV a b c

$$c(i,j)=a(i,j)/\max(1,b(i,j))$$

The maximum function in the denomimator replaces zero b values with unity (integer arithmetic is used) to avoid dividing by zero and still preserve essentially all of the brightness range of the image in b and therefore in c. For example a 0–255 brightness range is treated as if it were 1–255, a distortion that is not visible on any image (Russ, 1986). To date, there has not been a situation where this method of handling zero values in the divisor image has given problems.

##: IMAGE_BDR

Returns the bounding rectangle of the last image read into the system, e.g. (1 1 256 256) for quarter sized images.

##: IMAX a b c

$$c(i,j)=\max(a(i,j),\ b(i,j))$$

##: IMIN a b c

$$c(i,j)=\min(a(i,j),b(i,j))$$

##: IMUL a b c

$$c(i,j)=a(i,j)*b(i,j)$$

##: ISUB a b c

$$c(i,j)=a(i,j)-b(i,j)$$

##: LIMITS a
Returns the minimum and maximum of all $a(i,j)$

##: LOG a b 25

$$b(i,j)=\text{int}(25*\log(\min 1,(a(i,j))))$$

If any pixels in a are less than one, they are replaced with unity, and a message is sent to the terminal after the operation is complete. The logarithm of the a element is then taken to base ten, multiplied by a scale factor, and truncated to an integer. In this case the scale factor is 25. The scale factor can be real. It defaults to 50 if it is not given.

##: LVAL BLIST 2
This returns a list of properties of the second largest blob. The properties are described in the text.

##: LVAL a 100 100
This returns the value of a(100,100), the intensity of a single pixel.

##: MAG_NN a b 255 128 1.1 0.9
This command is used either as MAG_NN, which does nearest neighbour or zero order interpolation (Rosenfeld & Kak, 1982, p. 35), or as MAG_BILIN which does bilinear interpolation. This command example applies an x stretch of 10% and a y shrink of 10% to the image in a, centered around the coordinate (255,128). That is, most pixels are moved, but $a(255,128)$ at least, will be directly transferred to $b(255,128)$. For the nearest neighbour interpolation:

$$b(i,j)=a(255+(i-255)/1.1,128+j-128)/0.9)$$

The bounding rectangle applies only to b for this command, where usually it applies to either a or all arrays involved. If any coordinate calculated for a are out of the range of 1–512, then $b(i,j)=0$.
Formulas for the bilinear interpolation and other methods are given by Rosenfeld & Kak (1982, p. 35) or Pratt (1978, p. 113). These methods tend to smooth the images, giving a more pleasing visual effect. All interpolation methods degrade the image to some degree. The nearest neighbour interpolation will sometimes introduce intensity steps, while the bilinear interpolation has a slight blurring effect. We tend to use the nearest neighbour interpolation to avoid the blurring from the other interpolation methods.

##: MASK a b c 10 255

$$c(i,j)=a(i,j) \text{ if } 10 \leqslant b(i,j) \leqslant 255$$

$$c(i,j)=0 \text{ otherwise}$$

##: MEAN_FILTER a b w
$b(i,j)$ gets the mean value of all a pixels in the square centered on $a(i,j)$ that extends outward w pixels on each side of $a(i,j)$. This is a mathematically special case of CONVOLVE, written separately for convenience and speed—a sliding window is used, that is similar to the median filter algorithm (Huang *et al.*, 1979).

##: MEDIAN_FILTER a b w
$b(i,j)$ gets the median value of all a in the square centered on $a(i,j)$ that extends outward w pixels on each side of $a(i,j)$. An efficient algorithm is used (Huang *et al.*, 1979).

##: OUTLINE a b

$$b(i,j)=1$$

if $a(i,j)=1$ and any of the eight neighbours of $a(i,j)$ are zero, or if $a(i,j)$ falls on the bounding rectangle. The eight neighbours, include the four pixels at each corner of the pixel at (i,j).

##: PIXEL__MEAN a g h 4.0 i 10

This and the other two commands starting with 'pixel' are meant for use with a large number of small images, thus the letters denoting image arrays go beyond c. $h(i,j)$ is given the rounded mean value of $a(i,j)$, $b(i,j), \ldots, g(i,j)$ after the value has been multiplied by 4.0 and rounded to an integer. $i(i,j)$ is given the standard deviation (square root of the variance) of the same pixels after the value has been multiplied by 10 and rounded to an integer.

##: PIXEL__MEDIAN a h i m

The first two letters define the range of input arrays as with the PIXEL__MEAN command. The second two letters must include five arrays, which get the minimum value, the first quartile value, the median, the third quartile value and the maximum value, respectively.

##: PIXEL__T a 1 b 10 3 c 1 d 10 3 e 10

This command compares, pixel by pixel, the means of two groups of images (given by PIXEL__MEAN) with Student's t test. The pixel values are multiplied by the scale factors before performing the calculation or storing the t score. a and b are the arrays for the mean and standard deviations of group 1 and are followed by their scale factors and the number (three in this example) of images in the group. The next five parameters are the analogous values for the group 2 images. e receives the t value from the Student's t calculation, after being multiplied by the scale factor that follows (10 in this case).

##: RANK__FILTER a b w 90

$b(i,j)$ gets the 90 percentile value of all a in the square centered on $a(i,j)$ that extends outward w pixels on each side of $a(i,j)$.

Any percentile from 0 to 100 can be used. With percentile zero, this might be called a minimum filter, and with 100, a maximum filter (see Example 2).

##: ROTATE__NN a b 200 250 10·5

The interpolation options are the same as with the MAG__commands. This example places into b, the image in a, rotated counter-clockwise by 10·5° about the $a(200,250)$, using nearest neighbour interpolation.

$$b(i,j)=a(x,y)$$

where, replacing the centre point, $a(200,250)$, with $a(xc,sc)$, and replacing the angle, 10.5, with q:

$$i=\text{nint}((x-xc)*\cos(q)-(y-yc)*\sin(q)+xc)$$
$$j=\text{nint}((x-xc)*\sin(q)+(y-yc)*\cos(q)+yc)$$

For the coordinate transformation equations, see (Goldstein, 1950).

and where, since the bounding rectangle applies to $a(x,y)$, $b(i,j)=0$ whenever i or j falls outside the range 1–512

##: SCALE a b 200

The image in a is transferred to b with the dynamic range changed linearly so that the minimum value of all $b(i,j)$ is zero and the maximum is 200.

In general, let 200 be max__v, and let min__a and max__a, respectively, be the

minimum and maximum values for all $a(i,j)$. Then,

$$b(i,j)=(a(i,j)-\text{min\_a})*\text{max\_v}/(\text{max\_a}-\text{min\_a})$$

If max_v is not given, it defaults to 225 for display purposes. If all $a(i,j)$ are zero, the b is filled with zeros. If all $a(i,j)$ are the same value, then b is filled with max_v.

##: SDIV a b c
First, if $b(i,j)<1$, then $b(i,j)=1$.

Then, let max_v be the maximum value for all $a(i,j)/b(i,j)$. Then,

$$c(i,j)=\text{int}((a(i,j)/b(i,j))*255/\text{max\_v})$$

Zero values of the denominator are replaced with unity as with IDIV. The image in c is scaled to 255 for display.

##: SHRINK a b 4
This command is used to reduce the size and resolution of an image. The shrink factor can be any integer. For the case here, with a factor of four, each pixel in b is given the rounded mean value of the pixels in a in the 4×4 pixel square that the b pixel would cover if the b image were expanded about the lower left-hand corner of the image by a factor of four.

##: SLINE a 100 100 200 200
This command returns values to LISP from a that lie along the line from point (100,100) to point (200,200). The pixels are chosen as follows: Depending on which will have the greatest range, either i or j is stepped by one from the value in the first coordinate to the value in the second. The other index is calculated from the equation for a straight line with the two given coordinates as end points.

##: SQRT a b 30·5
This seldom used command gives the square root of the pixel intensities, rounded to the nearest integer.

$$b(i,j)=\text{int}(30{\cdot}5*\text{ sqrt}(b(i,j)))$$

A scale factor, 30·5 in this case, is used for avoiding loss of precision when taking square roots of smaller values.

##: THRESH a b 200 255

$$b(i,j)=1 \text{ if } 200\leqslant a(i,j)\leqslant 255$$

$$b(i,j)=0 \text{ otherwise}$$

##: TOP_HAT a b 2·5 4·6 10
This is a specialized gated filter for isolating compact objects (Bright & Steel, 1987). For a discussion on gated filters, see Rosenfeld (1984, p. 266). A template is generated using the three numbers—the radius of the inner circle (here 2·5 pixels), the radius of the outer circle (here 4·6 pixels) and a threshold value (10 intensity units). As with other window commands, a border is excluded. The border has width 5 in this case, as the outer circle will include pixels this distance away.

$$b(i,j)=a(i,j) \text{ if pass condition is met}$$

$$b(i,j)=0 \text{ otherwise}$$

The pass condition is that the maximum value of the (centres of the) pixels inside the inner circle, less the maximum value of the pixels outside this circle but inside the outer circle, must be greater than the threshold. The two circles are centred on the centre of the pixel. The indices of the neighbouring pixels are the positions of their

centres.

##: TRANS a b 300 400 200 200

This command translates pixels in the bounding rectangle of a to b:

$$b(i,j)=a(i-100,j-200)$$

where the four numbers, $x1$ $y1$ $x2$ $y2$, in the example are the example coordinate an destination coordinate. The coordinates need not lie in the bounding rectangle, b are used only to calculate the shifts. If the shifts make an index out of range, $b(i,j)=$

APPENDIX III. SAMPLE LISP FUNCTIONS

The first three examples are some of the LISP functions used to generate figures in th paper. These are good examples of image processing tasks. The power of the LISP languag is little used, and no image analysis is done—the functions are merely sequences o commands. The functions do serve to show how the commands are combined to achieve th desired images, and how the LISP functions can be organized to split the processing tas into meaningful parts. The fourth example is more complex, and shows some advantages o using LISP by using a few LISP functions to control the analysis.

The first line (starting with 'defun') always names the function and its input. An ope parenthesis followed immediately by a close parenthesis means that the function requires n input as with Examples 1 and 2. The lines that follow the first line usually consist of on call to a LISP function. The function calls are an open parenthesis, the function name, an input and a close parenthesis. Some of these calls are nested. Most of the function calls us 's' (for send), which takes the input and sends it to the FORTRAN LISPIX sub-system a a command. In other words, '(s '(IDIV a b a))' in a LISP function replaces typing 'IDIV b a' directly as a command. (Lispix is not case sensitive. The commands are printed i upper case for clarity.)

The third example, 'disp direction', has commas preceding some items as input to th 's' function. This is LISP syntax for substituting the value of a variable into the list bein sent as a command, for example if the variable 'scrt_ara' has the value 'b' and the variab 'angle' has the value 315, then

(s '(FILL ,scrt_ara ,angle))

in line 3 of the disp_direction function does the same thing as typing

FILL b 315

directly as a command.

For the first three examples, the LISP functions are written with the lines numbered fo explanation (normally LISP has no line numbers). The fourth example has comments and line labels following the semicolons. In contrast to the first three examples, LISPIX commands are on a minority of the lines due to the calculations and book keeping that LISP is required to perform.

1. *Sample LISP function to generate Fig. 3*

Typing (fig_example_3) as a LISP function call will generate Fig. 3. The function calls at the end that generate the annotation and control the display device are included only in this example, to show how they are used. Every use of the function 's' below invokes a LISPIX command. The commands in upper case are explained in the text. Those in lowe case are associated with a particular display device or with programming detail and are no covered in the text.

```
1 (defun fig_ example_3 ()
2    (s '(dir "sys$image:[brim.epma.wds.cr_8mar85]"))
```

```
3   (s '(get a cr400yc.dat))
4   (s '(BDR ,(s '(image_bdr))))
5   (s '(FILL b 62))
6   (s '(IDIV a b a))
7   (s '(MEDIAN_FILTER a b 3))
8   (s '(show 0 b))
9   (s '(llzoom 4))
10  (ll)
11  (label 'el _a2)
12  (s '(annotate 0 14 7×7 median filter)))
```

1. LISP beginning of function definition. The name of the function is fig_example_3, and it has no input parameters. Typing (fig_example_3) when in LISP will produce Fig. 3 on the screen and annotate it.
2. Sets the system default directory for the image file desired.
3. Reads in the image from the disc into image array a.
4. Reads the bounding rectangle for the image just read from the disc. In this case, it is 64×64. Then the default bounding rectangle is set to the size of the input image, so that all succeeding operations operate only on the image area rather than on the entire 512×512 arrays.
5. Generates an array with a constant value of 62. The maximum value of the image is 62 times too large to be displayed.
6. Divides the image, or scales it for display. The scale command is not used in this case because the scale command sets the minimum image pixel value to zero. Here, we desired the minimum pixel value to retain its true relative value.
7. Performs median filter on image in a and puts the result in b. The width parameter is 3, thus a pixel at (x,y) in the result image in b is the median of all of the pixels o
8. Display the result of the median filter operation.
9. Zooms the image from quarter size to full size on the display.
10. Generates a grey scale ramp for contrast enhancement. This a separate entire L function, using several specialized display commands that are not discussed in th paper.
11. Annotates the image display with the given label and other information including the time of day. This is another separate LISP function that again uses specialized display commands.
12. More display annotation giving a description of the image.

2. *Example LISP function to generate Fig. 44*

Typing (F44) as a LISP function call will generate the Fig. 44 in an image array. The commands that display the figure and annotate it have been removed for simplicity.

```
1 (defun F44 ( )
2   (s '(BDR))
3   (s '(FILL a))
4   (s '(circle_fill a 255 255 200))
5   (s '(DISTAMP a b 1 1))
6   (s '(copy b a))
7   (s '(GRADS a b c))
8   (disp_direction 'c 'b 'd 'e 1)
9   (s '(scale b c))
10  (get_graphics)
11  (s '(overlay c f d BW))
```

1. LISP definition of the function that generates the figure. Typing (F44) will generate the figure and put it in array d.
2. Resects the default bounding rectangle to the full image size. (Not necessary if system has just been started.)
3. Initializes array a to zero. (Not necessary if system has just been started.)
4. Generates a disc shaped mask with radius 200, centered in the middle of the image array.
5. Calculates the distance map for the disc. A mesh plot of this object (the values of the distance map) is a cone with the apex centered on the image.
6. Moves the object to array a, merely for convenience.
7. The direction of the gradient is put in c. This is the function that is to be displayed.
8. This is a separate LISP function, described below, that displays the direction of the gradient of the image generated in step 5, and stores the same result in array b for further manipulation.
9. Scales the display to full intensity (0–255 rather than 0–180)
10. Separate LISP function to generate the graphics: the heart shape and the axes. The graphics are put into array f for subsequent use by the overlay command, which will combine the graphics with the image to make the composite figure.
11. Combines the graphics in f with the image of the direction of the gradient in c for the final figure, left in d (which is actually displayed and photographed later). The BW option on the overlay command writes the graphics, as white with a one-pixel-wide black border, on top of the image. This border makes the graphics visible regardless of the shade of the underlying image.

3. *Example LISP function to scale values of direction in degrees to make an intuitive display*

Since this sequence of commands is useful for several projects, it is written as a function itself, and incorporated into the above function, line 8.

```
1 (defun disp_direction (in_ara out_ara scr1_ara scar2_ara)
2   (let* ((angle 315))
3   (s '(FILL ,scr2_ara ,angle))
4   (s '(ISUB ,in_ara ,scr2_ara ,out_ara))
5   (s '(THRESH ,out_ara ,scr1_ara -360 -1))
6   (s '(SCALE ,scr2_ara ,scr2_ara 360))
7   (s '(IADD ,scr2_ara ,out_ara ,out_ara))
8   (s '(FILL ,scr2_ara 180))
9   (s '(ISUB ,out_ara ,scr2_ara ,out_ara))
10  (s '(IABS ,out_ara ,out_ara))))
```

1. LISP definition of the function that takes the direction of the gradient (in in_ara, which has the letter designation of the array of that image), and generates the specially scaled version (in out_ara), and as side effects, destroys the contents of two scratch arrays (scr1_ara and scr2_ara).
2. Defines the angle corresponding to zero intensity. This particular value, 315°, makes the apparent illumination come from the upper left.

3–10. Computation steps to scale the image as described.

4. *LISP functions used in the automatic threshold selection algorithm, in Example 5 of the text*

Most image analysis problems are more complex than the above three examples in that the algorithm involves more than a fixed sequence of commands. Usually, information from

intermediate processing steps determines the subsequent steps and LISP lends itself to organizing that information. Further, the programming environment that LISP presents facilitates testing and debugging.

This fourth example demonstrates one of the simpler analysis functions in use, and is complex enough to show some advantages of using LISP. It required the writing of the three short LISP functions shown below. The example includes a few preprocessing steps followed by a search involving calculation of the distance map for different thresholds. Initially, the exam4 function determines the size of the image, and then a smaller representative central area in which to work. The gradient of the image is taken, and then a general binary search function is called. This search function is given the name of the width function which, in turn, uses a third function (twice, if necessary) to give LISPIX the distance map command and to tell the binary search function whether the current threshold is too large or too small. This third function keeps track of previous calculations to avoid repeating them. To write much of this procedure in other languages such as FORTRAN would be tedious at best.

The three LISP functions written for this example are shown below and are labelled to the right to correspond to the comments that follow them. Several general purpose functions are used by this code but are not shown.

The function exam4, the top level function in this example, is called by higher level functions that automatically select thresholds for images with sharply defined objects, as discussed in Example 5 in the text. Exam4 calculates the gradient (line B3) of such images so that edges now become the objects. Exam4 then repeatedly calculates the distance map for various thresholds to find the one at which, when decreased by one, the half-width of the widest object (edge) increases from one to two pixels. As mentioned in the text, this gives an adequate number of edge pixels for subsequent calculations while keeping the edges thin. For example, consider the magnitude of the gradient image in Fig. 22. At a threshold equal to zero all pixels are 'edge' pixels, while at a threshold equal to the maximum value of the gradient, only the brightest pixels are 'edge' pixels. The desired threshold lies between these extremes. The binary search function (Knuth, 1973) repeatedly cuts the threshold search range in half until it finds the desired threshold value.

```
(defparameter *wscr___tbl* nil                                    ;A
                "scratch space for storing DISTAMP results")

(defun exam4                                                      ;B
  "pick grad thresh ... chooses a threshold for narrow objects.
  Assumes image in array a. Uses arrays b, c and d.
  The test bdr is the centre area of the image."
  (let *((image_bdr (s 'image_bdr)) ;size of image                ;B1
         (dummy (s '(bdr ,image_bdr))) ;set default bdr to image size ;B2
         (dummy (s '(grada a b))) ;gradient of image into array b ;B3
         (edge (fourth image_bdr)) ;height (and width) of image, pixels
         (test_bdr (bdr_expand image_bdr (round edge -5)))        ;B5
         (max_grad_val (cadr (assoc 'max (s '(limits b))))))      ;B6
  (s '(copy b c)) ;keep b for display. Work on array c.
  (fill_bdr_outline 'c test_bdr 0)                                ;B8
  (setq *w_scr_tbl* nil) ;Initialize scratch area for x_w_ fun
  (binary_search '(0 ,max_grad_val)                               ;B10
                 1                                                ;B11
                 'width_function                                  ;B12
                 #'<)))                                           ;B13
```

```
(defun width_function (thresh)
  (let* ((v1 (x_w_fun thresh)) ;Max of distance map value for thresh.
         (v2 nil)) ;will be same as v1 but for (thresh -1).
  (cond ((> v1 1) 2) ;If V1 too large, return 2.
        (t (setq v2 (x_w_fun (1- thresh))) ;V1 may be ok. Get v2.
          (cond ((> v2 1) 1) ;If V2 ok, return 1.
                (t 0))))))) ;Otherwise, return 0.

(defun x_w_fun (threshold)
  (let* ((temp nil))
    (cond ((assoc threshold *w_scr_tbl*
             (cadr (assoc threshold *w_scr_tbl*)))
          (t (s "distmap c d threshold 32767)) ;Calculate distance map
            (setq temp (1- (cadr (assoc 'max (s "limits d)))))
            (setq *w_scr_tbl* (cons (list threshold temp) *w_scr_tbl*))
            temp))))
```

Notes on lines of LISP functions

A. This statement defines a scratch area (width scratch table) for temporary storage of results of the width function which may be used later (D2 and D3).

B. Exam4 is the top level function that returns a value for the threshold of the image in c. As mentioned in Example 5 of the text, this threshold is used to generate a mask (of edges) from the gradient image in c, which in turn is used to select edge pixels from the original image in a. The threshold for distinguishing the two metallic phases in image a is the average of the edge pixels thus selected.

B1. The LISP let* construct assigns values sequentially to the variables image_bdr, dummy, edge, test_bdr and max_grad_val. The value for 'dummy' is not used later, but the LISPIX commands BDR and GRADA need to be given before max_grad_val can be given the maximum value of the gradient.

B5. Bdr_expand is a function that expands or, in this case, contracts the boundary of a given bounding rectangle by the given amount of pixels. Round is a LISP function that gives a rounded value of the edge (dimension of the image) divided by 5. This algorithm presently works on the centre 64% area (each edge reduced by 20%) of the image to save time and avoid any blurring or distortion at the image boundaries.

B6. Max_grad_val is the maximum value of the gradient image within the default bdr given by bdr_expand above. The threshold being searched for by the exam4 function is between zero and this value.

B8. The distance map algorithm requires a boundary around all objects that is at least one pixel wide. To ensure this, the fill_brd_outline function (not shown) puts such a border around the default bounding rectangle with pixels of value zero.

B10. The value returned by exam4 is the value calculated by the binary search function (not shown). This is a general function that performs a binary search in one variable. For use here, it is given a range of values of the threshold in which to search (B10), the value of the width function for which a threshold is sought (B11), the name of the width function (B12), and the direction of change in the width function—it decreases (B13) with increasing threshold in this case.

C. *This width function, used by the binary search function above, gives the width of the* widest object in the gradient image, when the threshold is set to the value of the variable, thresh. The maximum width being sought is three pixels, that is a central pixel plus one on each side. The width function returns 1 if thresh gives this width and (thresh −1) gives a greater width. The width function returns 0 if thresh is too

large and 2 if thresh is too small. If necessary, this function calls the auxillary width function (D) twice (C1 and C4).

D. This is an auxilary width function that calculates the distance map of the gradient image in c at the given threshold. Due to the searching strategy of the binary search function and the width function, the value for this auxillary function may be required more than once for any given threshold. Since calculating such values is time consuming, they are stored in the scratch area, *w_scr_tbl*, along with the corresponding threshold (D6), as a list: ((threshold1 width1) (threshold2 width2) ...). If a width for any particular threshold is required later (D2) it is retrieved from the list using assoc, a built-in LISP function used here to search the scratch area using the threshold value as a key. The value returned by this auxillary function is, then, either a previously stored number (D2, D3) or a newly calculated number (D5, D7).